SPECIFICATION AMENDMENTS

Please amend the specification as follows:

Substitute the paragraphs [0040] and [0041] at page 16, with the following:

[001] Each of the user interface display elements (e.g., display elements 220, 222 ~~220~~, and the like) displayed in the list box 216 have a data context property that defines the data item 224 as the data source of the user interface elements. An additional binding definition 124 can be implemented to associate the element properties of the user interface display elements with additional data item properties of the data item 224. The user interface display elements displayed in list box 216 have a dependent association to the user interface display element 220 such that the additional binding definition 124 defaults to the data context property of the user interface display element 220 to define the data item 224 as the data source of the additional user interface elements.

[002] The data programming model 110 defines a data context property on all of the user interface display elements and the property value is the default object to be used as the source data item for all binding definitions 124 attached to a user interface display element. The data context property is an inherited property such that when a user interface element does not define a value for data context, it uses the ~~used~~ same property value as its parent. This provides a developer with a convenient way to define the source data item for an entire subtree of user interface elements.

Substitute the paragraphs [0043] at page 17, with the following:

[003] When an items control contains a large number of data items (i.e., more than can be displayed at one time on a user interface display), virtualization can be implemented to generate user interface display elements for only those data items that actually appear in the display. As a user scrolls through the list of data items, the items control, together with the associated generator, automatically generates the user interface elements to represent the data items that are currently visible, and discards the user interface elements for the data items that are no longer visible. This process may also be referred to as user interface virtualization because it pertains to keeping only a small number of virtual user interface subtrees. A second level of virtualization, referred to as item virtualization, pertains to a similar technique applied to the data items themselves. Rather than maintaining a full list of data items in memory, a virtualizing collection keeps only those data items for the user interface layout and the application program itself. The remaining data items are stored externally, in a disk file or a local database or even a remote database, and are only retrieved when needed.